REMARKS

Claims 1 to 2 were pending. Claim 1 has been amended and claims 3 to 26 have been added to by this Amendment. Applicant respectfully submits claims 1 to 26 for consideration. Support for the Amendment can be found throughout the specification and at least at page 39, line 0, to page 45, line 15. Applicant has attached a marked-up version of the changes being made by the current amendment.

The examiner objected to the drawings. As described above, figures FIG. 2, FIG. 3, FIG. 4A, FIG. 5, FIG. 6, FIG. 9, FIG. 10 and FIG. 12 have been substituted to correct typographical errors and/or omissions on the originally filed sheets. Specifically, in FIG. 2, the reference number 14 was changed to 15. In FIG. 3, the reference number 38 was changed to 38'. In FIG. 4A, the letters OT in block 88 were changed to TO. In FIG. 5, the reference number 82 was added. In FIG. 6, the reference number 16 was changed to 15 and the reference 36, (FIG. 3) was changed to 36, (FIG. 1). In FIG. 9, the reference number 360 was added. In FIG. 10, the reference 74, (FIG. 3) was changed to 374, (FIG. 9). In FIG. 12, the reference number 101 was changed to 401, the reference number 102 was changed to 402 and the reference number366 was deleted.

The examiner objected to the title. As described above, the title has been amended. The examiner notes typographical errors in the disclosure. As described above, the disclosure has been amended.

The examiner rejected claims 1 and 2 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Specifically, the examiner states that it was unclear whether claim 1, and thus dependent claim 2, were method or system claims. As amended, claim 1 is directed to a system. In light of the amendment, Applicant respectfully requests that the rejection be withdrawn.

The examiner objected to claim 2 as using the term "graph" in a way that is repugnant to the usual meaning. Applicant respectfully disagrees with such characterization. Using the term "graph" in connection with a diagram, for example FIG 3A, is an accepted practice in the art, as

evidenced by "Encyclopedia of Computer Science and Engineering, 663-668 (Anthony Ralston, Edwin Reilly, Jr eds., Van Nostrand Reinhold Company, 2d ed. 1983). In light of the use of this term in the art, Applicant respectfully requests that the objection be withdrawn.

The examiner rejected claims 1 and 2 under 35 U.S.C. § 102 as being anticipated by U.S. Patent No. 6,295,521 to DeMarcken et al. ("DeMarcken"). Claims 1 and 2 as amended distinguish over DeMarken since DeMarken neither describes nor suggests a selection module configured to output a set of diverse travel options smaller than a candidate set of travel options by selecting from the candidate set of options, for each travel requirement in the plurality, one or more travel options that satisfy that travel requirement, wherein the candidate set is represented by a compact representation.

The claimed invention generates a set of diverse travel options by generating a plurality of travel requirements and for each requirement, selects one or more travel options that satisfy the requirement to output a set of diverse travel options. For example, the selection module will include in this diverse set travel options for trips on Carriers A, B and C, non-stop trips, even if there are 100 trips that are not non-stop but cheaper, trips at all time periods throughout the day, and the like. In contrast, DeMarcken generates a list of travel options without regard to diversity. DeMarken enumerates travel options based on a value function, for example lowest price or shortest total duration. For example, DeMarcken will generate a list of travel options that include the top N number of least expensive trips meeting the user's entered parameters. Contrary to applicant's invention, this approach can result in a set of travel options with no trips on Carriers B and C, no non-stop trips and no trips at a certain time of the day, because they are not included in the top N number of least expensive trips.

Claim 2 depends from claim 1 and thus distinguishes from DeMarken.

The examiner rejected claim 1 under 35 U.S.C. § 102 as being anticipated by U.S. Patent No. 5,832,453 to O'Brien ("O'Brien"). Applicant's amended claim 1 distinguishes from O'Brien since O'Brien does not disclose a selection module configured to output a set of diverse travel options smaller than a candidate set of travel options by selecting from the candidate set of options, for each travel requirement in the plurality, one or more travel options that satisfy that travel requirement, wherein the candidate set is represented by a compact representation. O'Brien relates to determining a travel scheme minimizing travel costs for an organization. In

contrast to Applicant's invention, O'Brien does not disclose a selection module to output a set of diverse travel options from the candidate set of options. O'Brien is concerned with finding a solution with a minimum cost for an organization, not a solution that outputs a diverse set of travel options. Therefore, not only does O'Brien fail to disclose the Applicant's selection module, O'Brien cannot be seen to even teach or suggest it.

Moreover, O'Brien does not disclose a candidate set that is represented by a compact representation. As stated in the instant Application, the exemplary graph is a compact representation because the number of nodes needed to represent a typical pricing solution will be substantially less than the actual number of pricing solutions represented by the compact representation. As shown in O'Brien, some of the airlines do not fly some of the links, thus leaving empty cells. The table in O'Brien actually contains more cells than there are pricing solutions. Because the O'Brien representation is not smaller, but bigger than the pricing solution, O'Brien cannot even be seen to teach or suggest a compact representation. In light of the amendment, Applicant respectfully requests that the rejection be withdrawn.

The examiner rejected claim 2 under 35 U.S.C. § 103 as being U.S. Patent No. 5,832,453 to O'Brien ("O'Brien"). Applicant's claim 2 is distinct from O'Brien since does not teach or suggest that the candidate set is represented by a compact representation (claim 1). Accordingly, claim 2 which calls for the pricing graph is not suggested by O'Brien.

New claims 3-26 contain similar patentably distinct limitations and the arguments above are applicable.

Applicant respectfully requests that claims 1-26 as amended are now in condition for allowance. Enclosed are a $276 check for excess claim fees and a $920 check for the Petition for Extension of Time fee. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 7/2/07

David Miranda
Reg. No. 42,898

Fish & Richardson P.C.
225 Franklin Street
Boston, Massachusetts 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

20459858.doc

#9/a

Version with markings to show changes made

In the specification:

The paragraph beginning at page 4, line 27, has been amended as follows:

The system 10 also includes a plurality 30 of clients 30a-30c implemented by terminals or preferably personal computers. The clients 30a-30c are coupled to the server 12 via a network 22 which is also used to couple the remote resources (21a-21c) that supply the databases 20a-20b to the server 12. The network 22 can be any local or wide area network or an arrangement such as the Internet.

The paragraph beginning at page 5, line 2, has been amended as follows:

The clients 30a-30c are preferably smart clients. That is, using client 30c as an illustrative example, client 30c includes a client computer system 32 including a computer memory or storage media [34] that stores a client process 36 and a set of pricing solutions 38. The set of pricing solutions 38 in one embodiment is provided from the server process 16 and comprises a set of fares that are valid for a journey, and associated information linking the fares to the light segments of the journey.

The paragraph beginning at page 5, line 24, has been amended as follows:

Referring now to FIG. 2, the server process [18] 15 is preferably executed on the server computer 12 but could be executed on the client computer 32. The server process [18] 15 is responsive to the user input query 48. The user input query 48 would typically include minimal information needed to determine a set of pricing solutions. This information typically requires at a minim, an origin and a destination for travel. In addition, the information could also include times, dates and so forth.

The paragraph beginning at page 6, line 1, has been amended as follows:

This query 48 is fed to the scheduler process 16 that produces a large number of itineraries, that is, sequences of flight segments between the origin and destination for each slice

of a journey. Examples of scheduler systems that may be used include the OAG Flight Desk (Official Airlines Guide, a division of Reed Travel Group) or schedule components of computer reservation systems (CRS'[=]s) such as Sabre7, Apollo7, Amadeus7 and WorldSpan7. It is preferable in order to obtain the largest number of possible itineraries to use a scheduler with dynamic connection generation. Such a scheduler is described in co-pending patent application entitled SCHEDULER SYSTEM FOR TRAVEL PLANNING SYSTEM, Serial No. 09/109,622, filed on July 2, 1998 by Carl G. deMarcken et al. and assigned to the assignee of the invention and incorporated herein by reference.

The paragraph beginning on page 6, line 27, is amended as follows:

The set of pricing solutions 38 is used by an availability system 58 that interrogates an airline inventory database 20b to determine whether there are seats available on particular flights for particular pricing solutions. The availability system 58 uses the airline inventory database 20b as a filter to remove from the set of pricing solutions 38 those pricing solutions for which there are not available seats. The availability system 58 is shown after the faring process 18. However, it could be included at nearly any point in the server process [18] 15. In addition, it is shown being fed by the pricing solution when it may only receive flight information from the scheduler process 16 depending on the airline.

The paragraph beginning on page 7, line 7, is amended as follows:

The client system 30c receives the results from the server process [18] 15. These results are the set of pricing solutions 38 and/or pricing solutions based upon availability. The client process 36 executed in the client 30c uses this information or a subset of it to access a booking system 62 to provide a booking and reservation for a user selected, enumerated pricing solution, as will be described below.

The paragraph beginning on page 8, line 5, is amended as follows:

The client process 36 receives the flight information from scheduler process 16 and the pricing solution from the faring process 18 or the availability system [56] 58 and enumerates pricing solutions from the directed acyclic graph (DAG) representation. The

enumerated set of pricing solutions is rendered in a graphical user interface 41 on the client monitor 40 (FIG. 1) in a manner as will be described below.

The paragraph beginning on page 8, line 12, is amended as follows:

In response to user input 76, the client [40] 30c can manipulate travel options and can query the local copy of the DAG to produce and display a subset of pricing solutions enumerated from the DAG that satisfy the query 76. The manipulation process used to control the display and change the travel options will be described below.

The paragraph beginning on page 8, line 18, is amended as follows:

A directed acyclic graph (DAG) is used to represent the compact set of pricing solutions 38' since, in general, the number of nodes needed to represent a typical pricing solution will be substantially less than the actual number of pricing solutions represented by the DAG. This significantly increases the efficiency of transfer of a set of pricing solutions 38 from the server process [18] 15 to the client process 36. The DAG representation also minimizes the storage requirements for the set of pricing solutions 38. The DAG representation permits the use of powerful search, sorting and manipulation processes to produce various subsets of set of pricing solutions in an efficient manner. As used herein, a directed acyclic graph (DAG) is a set of *nodes* connected by directed arcs, that have no loops of arcs in the same direction. If a node A is connected to a node B via an arc A6B, then A is called a parent of B, and B is called a child of A. Each node may have zero, one or many parents and zero, one or many children. As used herein, a pricing solution that is represented by a graphy will be referred to as a pricing graph.

The paragraph beginning on page 10, line 29, is amended as follows:

The set of pricing-solutions [that] represented in the pricing-graph is presented in TABLE 2 below.

The paragraph beginning on page 15, line 14, is amended as follows:

After the faring process 18 decomposes the itineraries into faring atoms, the faring process 18 retrieves fares 84 and rules 86 for each faring atom by accessing the fares/rules

database 20a mentioned above. At this point a fare[=]'s routing is retrieved from a routing database and applied to a faring atom. If the routing test fails, the fare cannot be applied to the faring atom and a fare component is not built.

The paragraph beginning on page 15, line 21, is amended as follows:

The faring process 18 applies the rules 88 to the faring atoms to produce fare components. Fare-components are combinations of faring-atoms and fares. Fare-components (TABLE 6) are produced if a fare[=]'s rules pass a preliminary check on a faring-atom. They are used to store deferred rules (e.g., deferred record-2s and combinability record-2s) that are applied at a later stage of processing. Fare components also store extra information produced during the rule-checking process, such as information about surcharges and penalties and discounts that are applied to the base fare price.

In TABLE 6 on page 16, please substitute " ' " for all occurrences of "=".

The paragraph beginning on page 16, line 10, is amended as follows:

From the fare components the faring process 18 constructs 90 priceable units. For certain types of rules such as those which require access to fares and/or flights from outside of the fare component, those rules are stored 88a in the fare component for later or deferred evaluation. The priceable unit process 90, takes valid fare components and constructs priceable units from the fare components. This process 90 involves grouping fare components from different slices and checking fare component combination restrictions. At this stage of processing, the rules deferred in steps 88 and 88a are reapplied.

The paragraph beginning on page 17, line 14, is amended as follows:

After evaluation of the deferred record-2s at the priceable unit stage, the itineraries and priceable units are grouped together into a complete set of pricing solutions. This occurs by a link process 94 that links itineraries to corresponding pricing units from different slices to provide the pricing solution. At this juncture, any remaining cross-priceable unit fare combinability checks are performed to eliminate invalid combinations.

The paragraph beginning on page 18, line 26, is amended as follows:

The pricing solution resulting from the linking process 94 is used to construct 96 a pricing graph from the various data structures built during the preceding processes. This pricing graph is transmitted to the client process or can be stored for later use or transmission. A pseudocode representation of the high level processing logic involved in the above search procedure is set out below in TABLE 11.

The paragraph beginning at page 19, line 5 has been amended into two paragraphs as follows:

Referring now to FIG. 5, the process 82 to decompose an itinerary into faring atoms includes a retrieval process 100 that retrieves all itineraries for all slices in a journey. For each itinerary in each slice, the process 82 groups fairing atoms by faring markets at 104 and partitions itineraries into the divisions of faring atoms at 106.

[a] A pricing graph 38' (FIG. 3) is produced containing logically combinable nodes that can be used to efficiently and compactly represent a set of pricing solutions 38 (FIG. 1). The pricing graph 38' as used herein is a so-called directed acyclic graph although other types of representations could be used. For example, a grammar could be used.

The paragraph beginning on page 20, line 5, is amended as follows:

The pricing graph 38' is constructed [300] from data structures [302] (summarized below in TABLE 12 and mentioned in conjunction with FIGS. 4A-4B) provided during the preceding processes. The data structures convert [304] to one or more nodes in the pricing graph. The open-label-set data structure becomes an OR node and its children are the converted versions of its backward links. Each backward link in the open label set is converted to an AND node. If a pricing object is shared, for example, a priceable unit core is shared between several priceable unit labels, then its counterpart nodes in the pricing graph will be shared so that the size of the pricing graph is not unnecessarily enlarged. The converted nodes are placed [306] in the pricing graph nodes. Terminal objects such as fares and itineraries undergo essentially no conversion. They are placed [308] into special terminal nodes with no children and are obtained

from the various data structures that have the pricing objects.

The paragraph beginning on page 23, line 3, is amended as follows:

The pricing graph 38' resulting from the search procedure [54] provides a compact way for representing a very large number of set of pricing solutions. By the above process, it is often possible to obtain a very large number of pricing solution components. Although the number of pricing solutions can be returned in-the form of a simple list, this is not desirable, as a very large number of pricing solutions can be difficult to manipulate, enumerate and interrogate and to transfer/transmit across a network since the amount of data involved is very large. The pricing graph 38' provides a more compact way of representing these pricing solutions. The compact representation of the range of set of pricing solutions is generated where choices are represented explicitly and redundancy is removed wherever possible.

The paragraph beginning on page 24, line 4, is amended as follows:

As mentioned above, the pricing graph 38' produced by the search procedure [54] includes three types of nodes. The first type of node is a node that represents choices called "LOGICAL OR" nodes. The second type of node is a node that represents collections referred to as "LOGICAL AND" nodes. A third type of node represented in the pricing graph is a terminal node that represents pricing objects.

In TABLE 14, on page 24, please substitue " ' " for all occurrences of "=".

The paragraph beginning on page 25, line 9, is amended as follows:

Referring now to FIG. 6, a high level illustration of a process 300 that operates on the pricing graph 38' typically as a client process 36 on the client computer system is shown. The process 300 includes a user query 302 that passes parameters into a process [304] 306 and a value function [306] 304 to extract from the pricing graph 38' certain pricing solutions [308] that satisfy parameters specified by the user query 302. The extracted pricing solutions are returned as a list [308] or other representation. Generally the pricing solutions are displayed on the display 40. Display software [309] handles production of GUI[=]'s 41 to present the information.

The paragraph beginning on page 25, line 20, is amended as follows:

The pricing solution list [308] will contain pricing solutions extracted from the pricing graph 38'in accordance with user specified parameters from the user query 302 using one of the processes 304 [and one] of FIG. 7, where some exemplary illustrative processes are shown. In particular, in response to the user query 302, one of the processes is executed. The processes 304 can comprise a find best "value" and pricing solutions associated with the value (e. g., price) process 304a; find best "value" and pricing solutions associated with the value for "node" (e. g., find best price for a particular itinerary) process 304b; an enumeration for "N" pricing solutions 304c; or an enumeration process that lists the pricing solutions according to some "value." Additional enumeration processes can be provided to produce query results corresponding to different ways of looking at pricing solutions extracted from the pricing graph 38'. A node invalidating process [304e] (not shown) that invalidates selected nodes from contributing to a pricing solution is also included.

The paragraph beginning on page 26, line 7, is amended as follows:

Efficient algorithms 304 are used for manipulating this representation to extract information of interest and to enumerate set of pricing solutions from the structure. For example, it is possible to quickly extract the cheapest solution; to find the cheapest solution involving selected fares and itineraries; to verify whether any pricing solution remains if specific fares or itineraries are excluded; to enumerate solutions under various orderings and so forth. Furthermore, the representation is compact enough so that it can be efficiently stored and transmitted such as from the server to the client. One benefit, therefore, is that after a single fare search [54] in the server process [16] 15, the server process [16] 15 transfers the pricing graph 38 to the client process 36. The client process 36 can examine and manipulate the large space of pricing solutions represented in the pricing graph 38' without further interaction with the server process 18.

The paragraph beginning on page 26, line 23, is amended as follows:

For the set of pricing solutions represented by the pricing graph 38'to be useful,

processes are provided to extract pricing solutions from the graph and manipulate the set of pricing solutions, as depicted in FIG. 7. In general, each of the enumeration processes to be described operate on the pricing graph to extract pricing solutions from the pricing graph according to particular criteria that can be set, for example, by a client system 30c (FIG. 2) in response to a user query 48 (FIG. [4] 2). Examples of user queries as well as a display representation for information extracted from the pricing graph will be described below.

The paragraph beginning on page 27, line 21, is amended as follows:

There are many processes or operations on the pricing graph 38' that use a *value-function,* a function that operates on the terminal nodes of the pricing graph 38'and returns a numerical value that can be used to rank pricing-solutions. Examples of 25 value-functions include *price* computed by summing the prices of fares (and penalties and surcharges) in a pricing-solution, duration, or convenience (that might be a mix of total travel-time with penalties for stops and airline-changes, for example), or mixes of each.

The paragraph beginning on page 32, line 2, is amended as follows:

Another procedure 304b finds, for each node, the best (i. e., minimum) value of some value-function over all the set of pricing solutions involving that node. Price function [306a] 304b finds for each itinerary, the cheapest price of any pricing solution that contains that itinerary. These values can be computed efficiently, if the value-function can be decomposed into a node-value-function.

The paragraph beginning on page 32, line 9, is amended as follows:

The best price value function [306a] 304b computes inner-values, as above, and computes for every node, an outer-value, equal to the minimum value contributed by all parts of the graph except that represented by the node. For each node, the minimum value of the value-function over all solutions that involve the node, (i.e., the total-value) is computed as the sum of that node's inner-value and outer-value.

The paragraph beginning on page 33, line 3, is amended as follows:

It may be desirable to "invalidate" [304e] certain nodes from the pricing-graph 38'. For example, itineraries containing or not containing specified airlines could be marked as not participating in the above algorithms, enabling the algorithms to find the best solutions involving or not involving these itineraries. The above algorithms can be easily adapted to accommodate checking whether the node is valid. In particular, the computation of inner-values, the first step in all the above algorithms, is modified to mark for every node whether the node represents any valid partial pricing-solutions given a specific query parameter. This information can be used in the rest of the algorithms. Every terminal node contains a field "valid?" that is either true or false. The compute-inner-values procedure uses these values to set the "valid?" field for non-terminals. See TABLE 17 below:

The paragraph beginning on page 41, line 7, is amended as follows:

The diversity process 360 thus includes a procedure for generating a diverse list of (N) travel options (Rts) from a larger list of travel options (Ts), that are the best travel options for a set of travel requirements (R), as defined by an ordering function F. The diversity process 360 generates 362 an prioritized (ordered) list of requirements Rs, and sorts 364 the list of travel options (Ts) by function (F) to produce a best-first ordered list (Ts2). The diversity process 360, initializes the list of result travel options (RTs) to be empty. If the remaining list of requirements (Rs) is empty, the process 360 returns an ordered list of diverse travel options (Rts). Otherwise, the diversity process selects 366 the first travel requirement (R) from the ordered list of requirements (Rs) and removes 368 a requirement (R) from the requirement list (Rs). The [diverstiy] diversity process 360 finds 370 a first (e.g., best) option T in the best-first ordered list (Ts2) that satisfies travel requirement (R).

The paragraph beginning on page 41, line 24, is amended as follows:

[If] In some embodiments the set of travel options is represented by a data structure that stores a large set of travel options by representing permitted combinations of smaller travel option components such as airline flights and fares. In such cases the travel option selection process above is implemented using a more complicated operation than searching through an

ordered list. With the pricing-graph the process for finding 370 the best travel option that satisfies a travel requirement is implemented for a representation that expresses travel options in terms of permitted combinations of smaller travel option components by disabling option components inconsistent with the requirement.

The paragraph beginning on page 42, line 3, is amended as follows:

As shown in FIG. 9A, the process 360 can use the node invalidating functions to invalidate nodes 422 in the pricing graph 38 that are inconsistent with the requirements. The process 360 [applys] applies 424 an enumeration algorithm that extracts the best solution from the pricing graph representation. The diversity process 360 calls an enumeration function, as described above, to enumerate all of the valid pricing solutions from the pricing graph that are remaining after the diversity process selectively invalidated nodes in the graph inconsistent with the travel requirements.

The paragraph beginning on page 42, line 13, is amended as follows:

If no option in the best-first ordered list (Ts2) satisfies [72] 372 the requirement (R), the process 360 goes to check 374 if the remaining list of requirements (Rs) is empty. Otherwise, the diversity process determines 376 if a travel option T is not already in result travel options list (RTs). If the option T is not in the list (RTs), the diversity process adds 378 the travel option T to end of the result travel option list (RTs). If the size of the travel option list (RTs) is equal to or greater than N 380 the process returns the ordered list of diverse travel options.

The paragraph beginning on page 42, line 23, is amended as follows:

Referring to FIG. 10[A], the diversity process 360 could optionally determine 382 for a travel requirement (R2)in the set of travel requirements (Rs), whether the requirement (R2)is included in a prior requirement (R), and whether the travel option T also satisfies 384 the requirement (R2). If the travel option T satisfies the requirement (R2), the process 360 can remove 386 the requirement R2 from the requirement list (Rs) and return to determine [74] 374 (FIG. [3] 9) if the remaining list of requirements is empty.

The paragraph beginning on page 43, line 19, is amended as follows:

The large candidate set of travel options may be analyzed 394 to find all parameters e. g., airlines found in any travel option, all departure dates for outbound and return, and all departure parts-of-day (morning, afternoon, evening) for outbound and return. The ordered list of requirements is generated by filling [96] 396 in for each template all airlines, dates and parts-of-day present in the options.

The paragraph beginning on page 44, line 1, is amended as follows:

Referring to FIG. 12, an alternative diversity process [100] 400 to generate a diverse set of travel options from a larger set of travel options is shown. The alternative diversity process 400 generates the best one or more travel options as defined by each of a set of different travel preference functions. The alternative diversity process 400 defines a set of travel preference functions with each function capable to order travel options. In one example, the set might include "cheapest", "quickest", and "most-convenient" where each is a function that assigns a numerical score to a complete travel option (such as price, total-trip-time, and total trip-time with penalties for stops). Functions that assign numerical values based on combinations of cost and convenience are possible, such as functions that weigh both price and time.

The paragraph beginning on page 44, line 15, is amended as follows:

Given set of travel options Ts 401, a set of preference functions Fs, and a desired number of answers for each preference function Ns, the alternative diversity process 400 returns a reduced set of diverse travel options Rts. The alternative diversity process initializes 402 a list of result travel options RTs to be empty and for each preference function F in the set of preference functions Fs and number of travel options (N) in the set of desired number of answers in each preference function (Ns), the alternative diversity process 400 computes 404 the N best travel options in Ts as defined by F. For each travel option T, unless the travel option T is in the set of diverse travel options Rts 406, the alternative diversity process 400 adds 408 the travel option T to the set of diverse travel options Rts and checks 410 the number of options. The alternative diversity Process 400 outputs 412 the diverse set of travel options (RTs).

The paragraph beginning on page 44, line 30 is amended as follows:

The diversity process can be run more than once with different travel option preference functions (a set of F's). For example, a travel planning system may wish to output a diversity of travel options that include diverse options that are cheap and [diverse] diverse options that are convenient, reflecting uncertainty in whether a [traveller] traveler is cost-sensitive or convenience-sensitive.

In the claims:

Claim 1 has been amended as follows:

1. A travel planning system comprising:

a requirements generator module configured to generate a plurality of travel requirements;

a selection module configured to [that] output[s] a set of diverse travel options smaller than [the complete] a candidate set of travel options [it has computed] by [pruning] selecting from the [larger] candidate set of options, for each travel requirement in the plurality, one or more travel options that satisfy that travel requirement [to a smaller set with a diversity-based pruning method], wherein the [larger] candidate set is represented by a compact representation.

In the title:

[A METHOD FOR] GENERATING A DIVERSE SET OF TRAVEL OPTIONS.

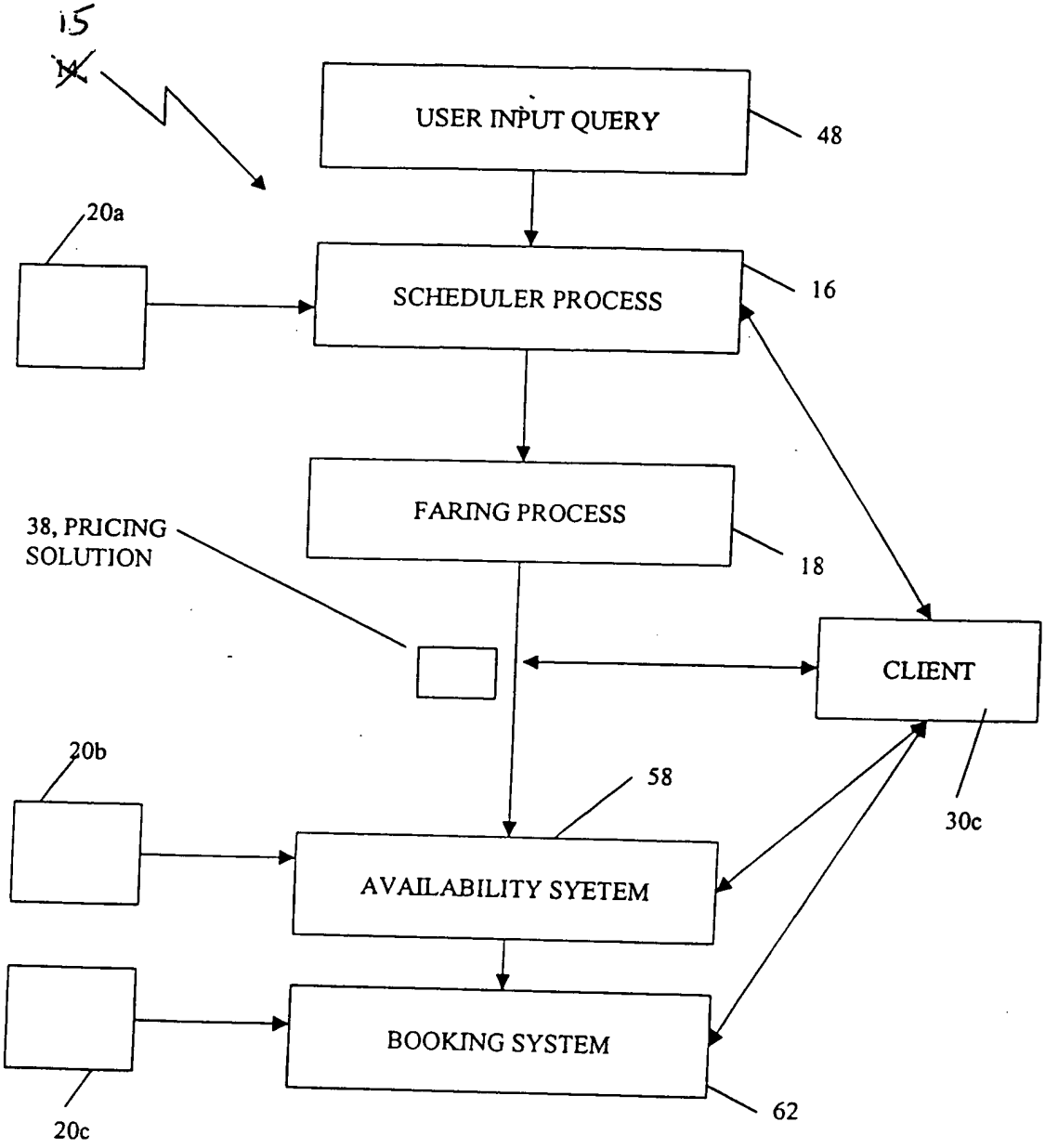
15
USER INPUT QUERY
48
20a
SCHEDULER PROCESS
16
FARING PROCESS
18
38, PRICING SOLUTION
CLIENT
20b
58
30c
AVAILABILITY SYETEM
BOOKING SYSTEM
62
20c

FIG. 2

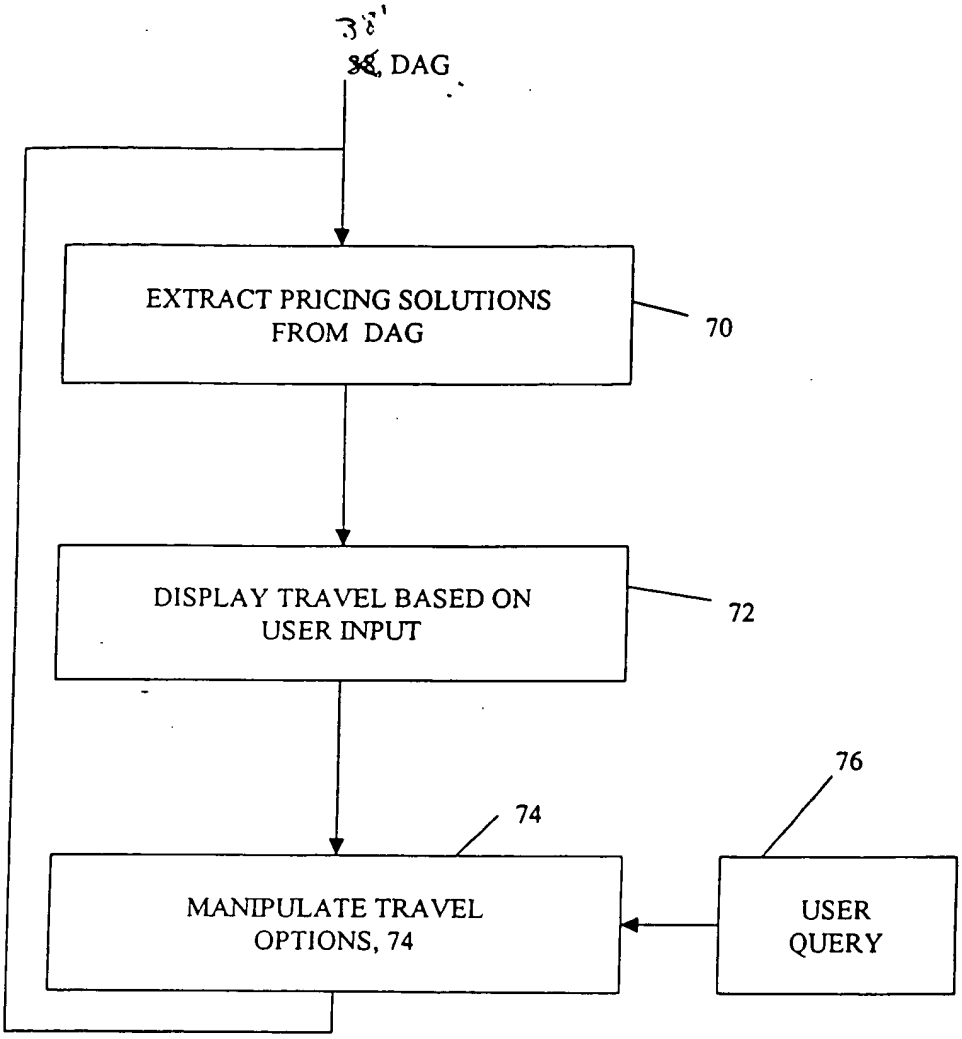
38, DAG
EXTRACT PRICING SOLUTIONS FROM DAG
70
DISPLAY TRAVEL BASED ON USER INPUT
72
74
76
MANIPULATE TRAVEL OPTIONS, 74
USER QUERY

FIG. 3

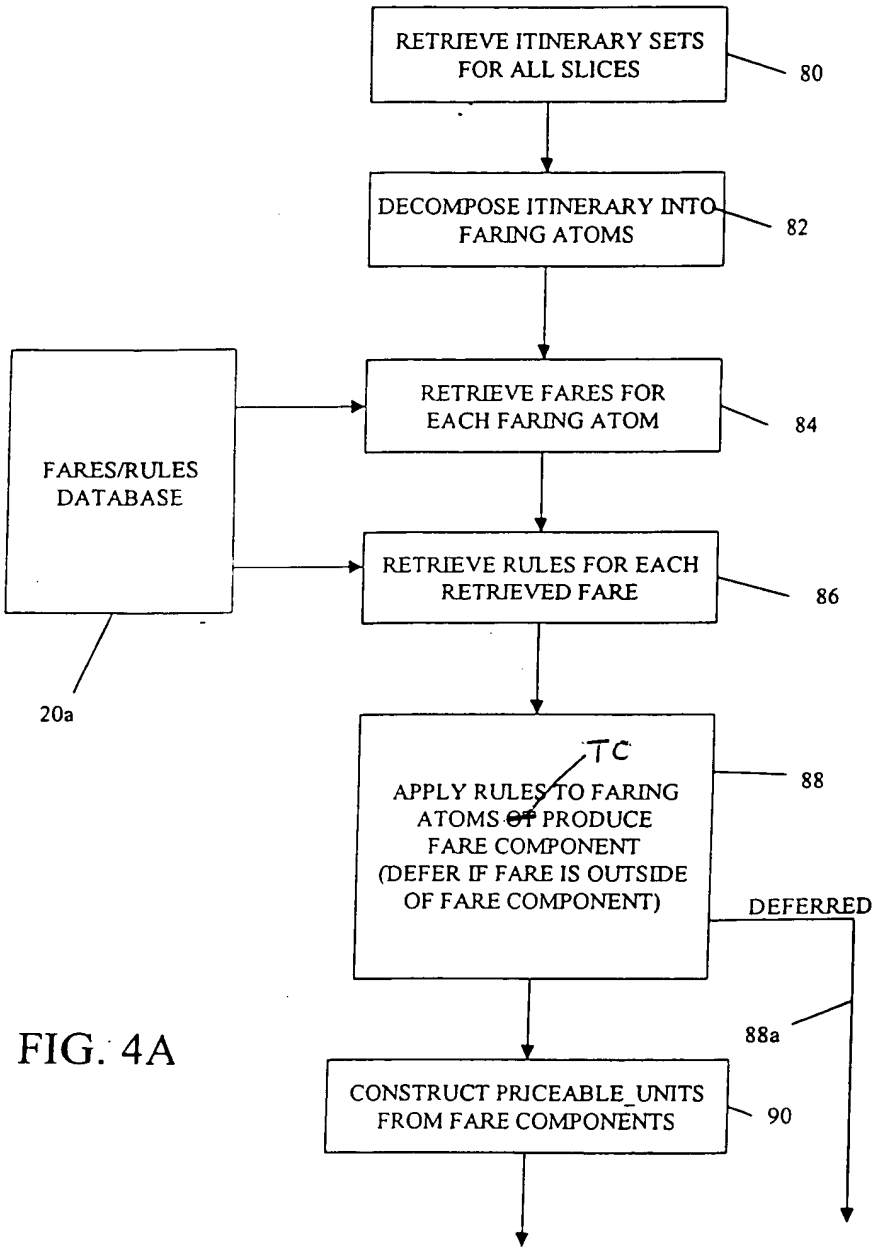
RETRIEVE ITINERARY SETS FOR ALL SLICES
80
DECOMPOSE ITINERARY INTO FARING ATOMS
82
FARES/RULES DATABASE
RETRIEVE FARES FOR EACH FARING ATOM
84
RETRIEVE RULES FOR EACH RETRIEVED FARE
86
20a
TC
APPLY RULES TO FARING ATOMS ~~OF~~ PRODUCE FARE COMPONENT (DEFER IF FARE IS OUTSIDE OF FARE COMPONENT)
88
DEFERRED
88a
CONSTRUCT PRICEABLE_UNITS FROM FARE COMPONENTS
90

FIG. 4A

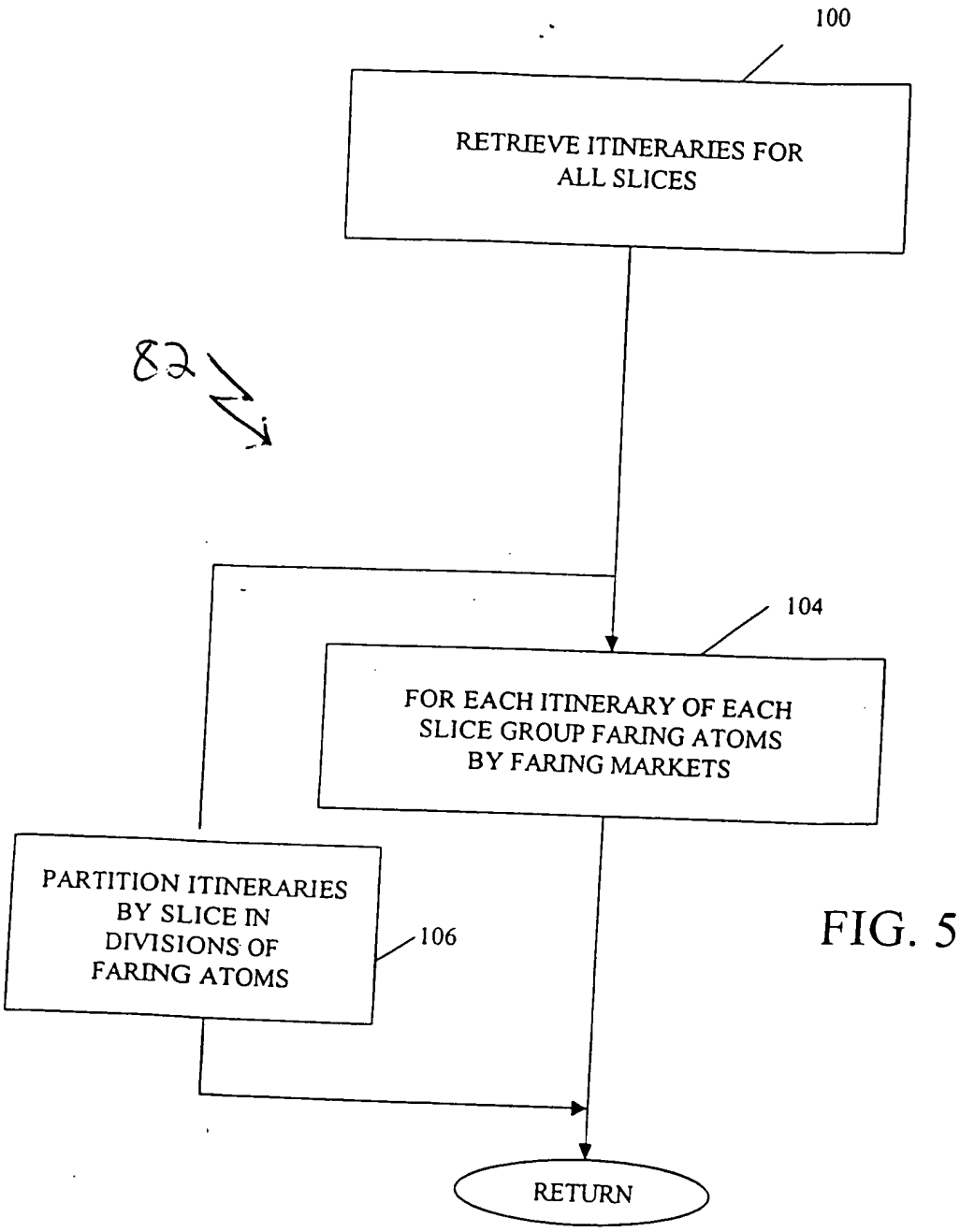
100
RETRIEVE ITINERARIES FOR ALL SLICES
82
104
FOR EACH ITINERARY OF EACH SLICE GROUP FARING ATOMS BY FARING MARKETS
PARTITION ITINERARIES BY SLICE IN DIVISIONS OF FARING ATOMS
106
RETURN

FIG. 5

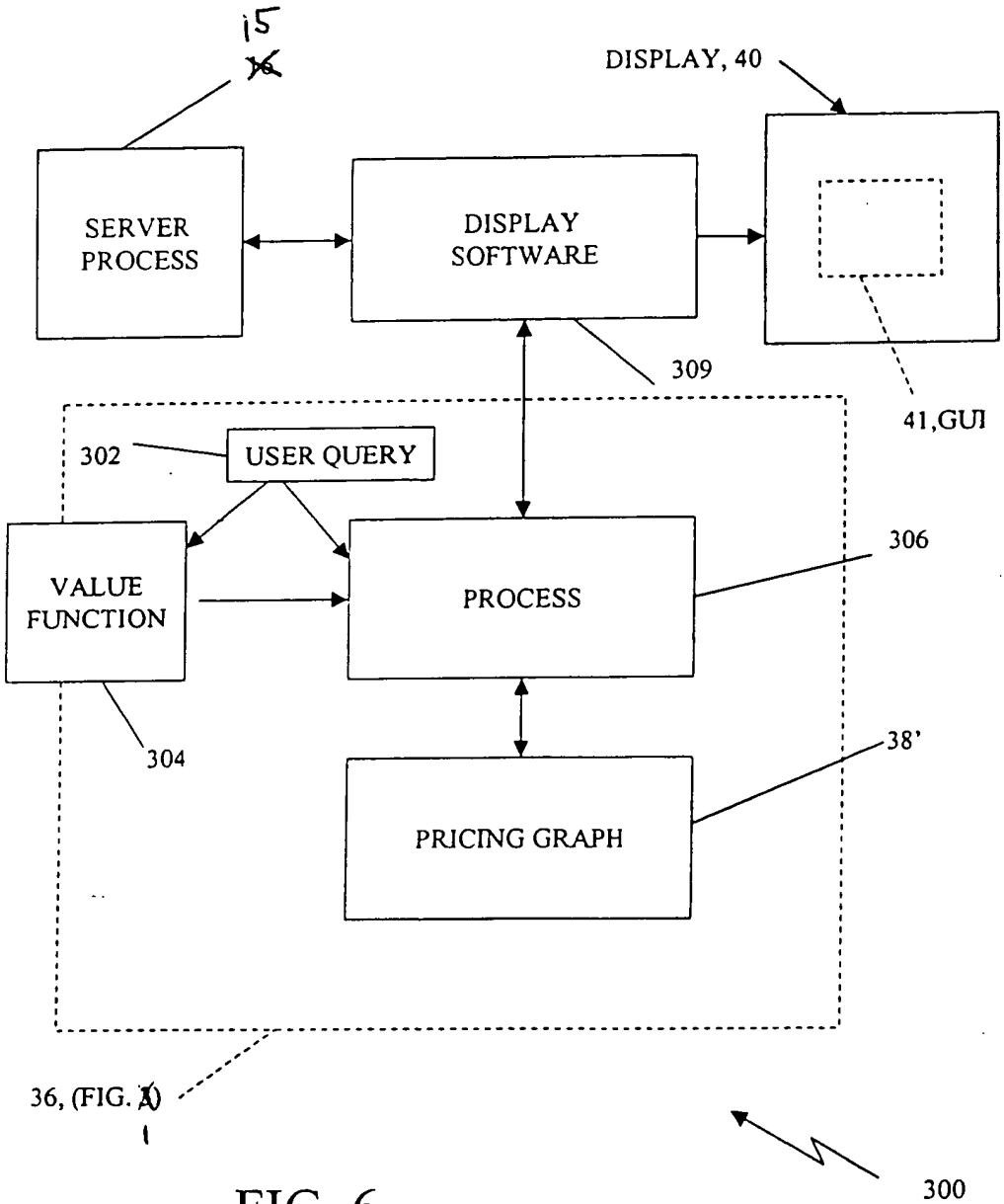
15
DISPLAY, 40
SERVER PROCESS
DISPLAY SOFTWARE
309
41,GUI
302
USER QUERY
VALUE FUNCTION
PROCESS
306
304
38'
PRICING GRAPH
36, (FIG. 1)
300

FIG. 6

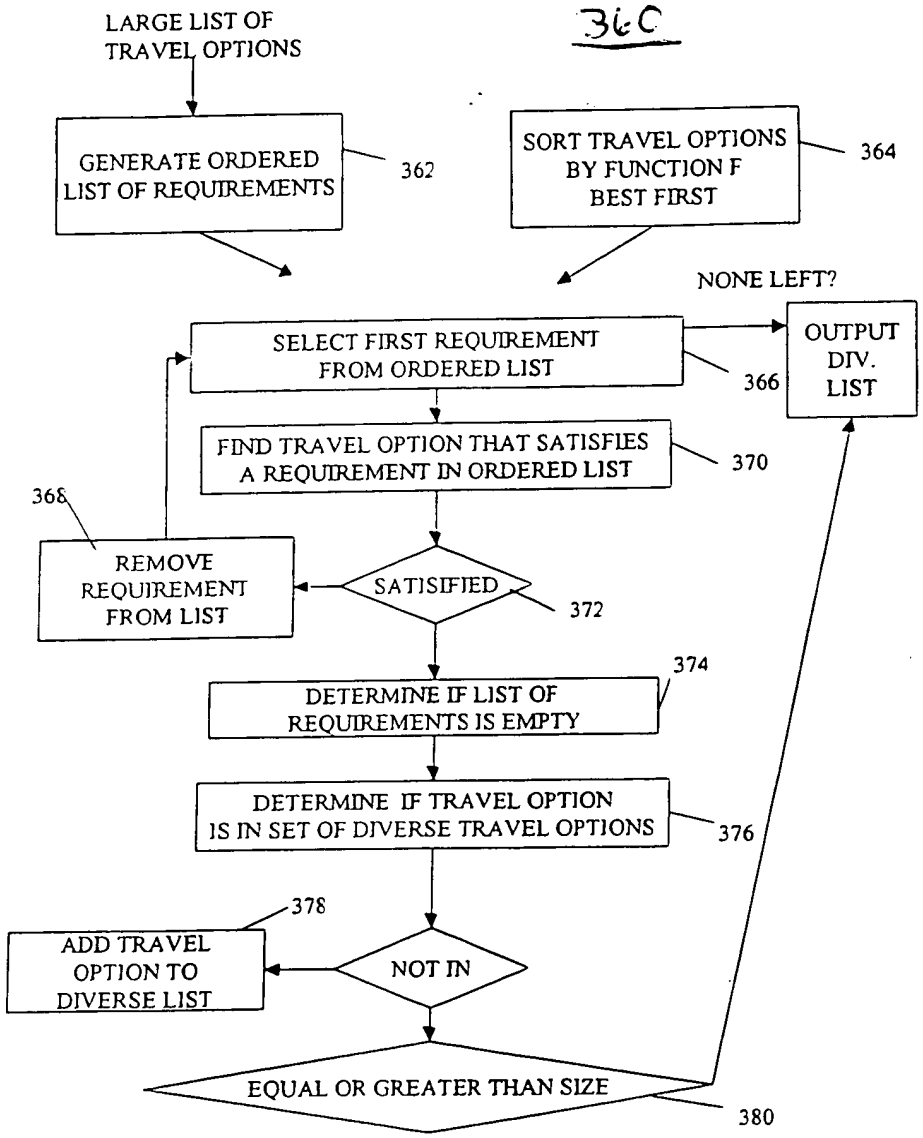
LARGE LIST OF TRAVEL OPTIONS
360
GENERATE ORDERED LIST OF REQUIREMENTS
362
SORT TRAVEL OPTIONS BY FUNCTION F BEST FIRST
364
NONE LEFT?
SELECT FIRST REQUIREMENT FROM ORDERED LIST
366
OUTPUT DIV. LIST
FIND TRAVEL OPTION THAT SATISFIES A REQUIREMENT IN ORDERED LIST
370
368
REMOVE REQUIREMENT FROM LIST
SATISIFIED
372
374
DETERMINE IF LIST OF REQUIREMENTS IS EMPTY
DETERMINE IF TRAVEL OPTION IS IN SET OF DIVERSE TRAVEL OPTIONS
376
378
ADD TRAVEL OPTION TO DIVERSE LIST
NOT IN
EQUAL OR GREATER THAN SIZE
380

FIG. 9

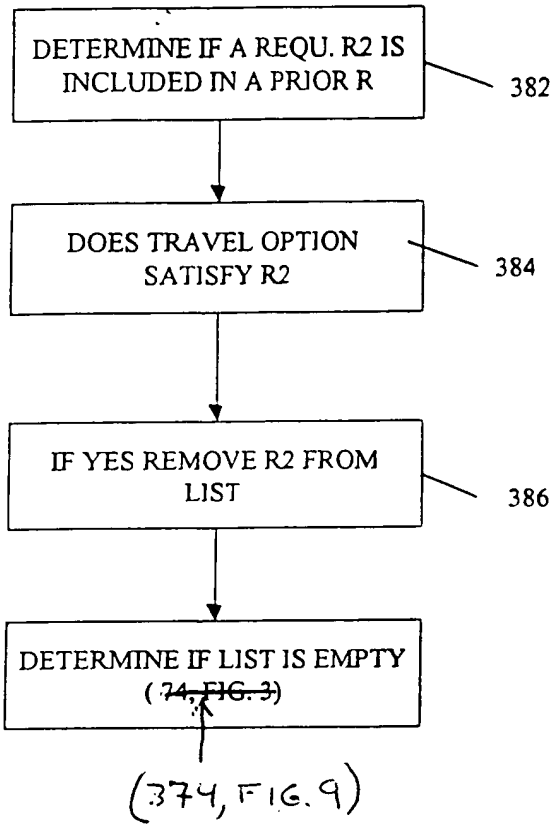
DETERMINE IF A REQU. R2 IS INCLUDED IN A PRIOR R
382
DOES TRAVEL OPTION SATISFY R2
384
IF YES REMOVE R2 FROM LIST
386
DETERMINE IF LIST IS EMPTY
(~~74, FIG. 3~~)
(374, FIG. 9)

FIG. 10

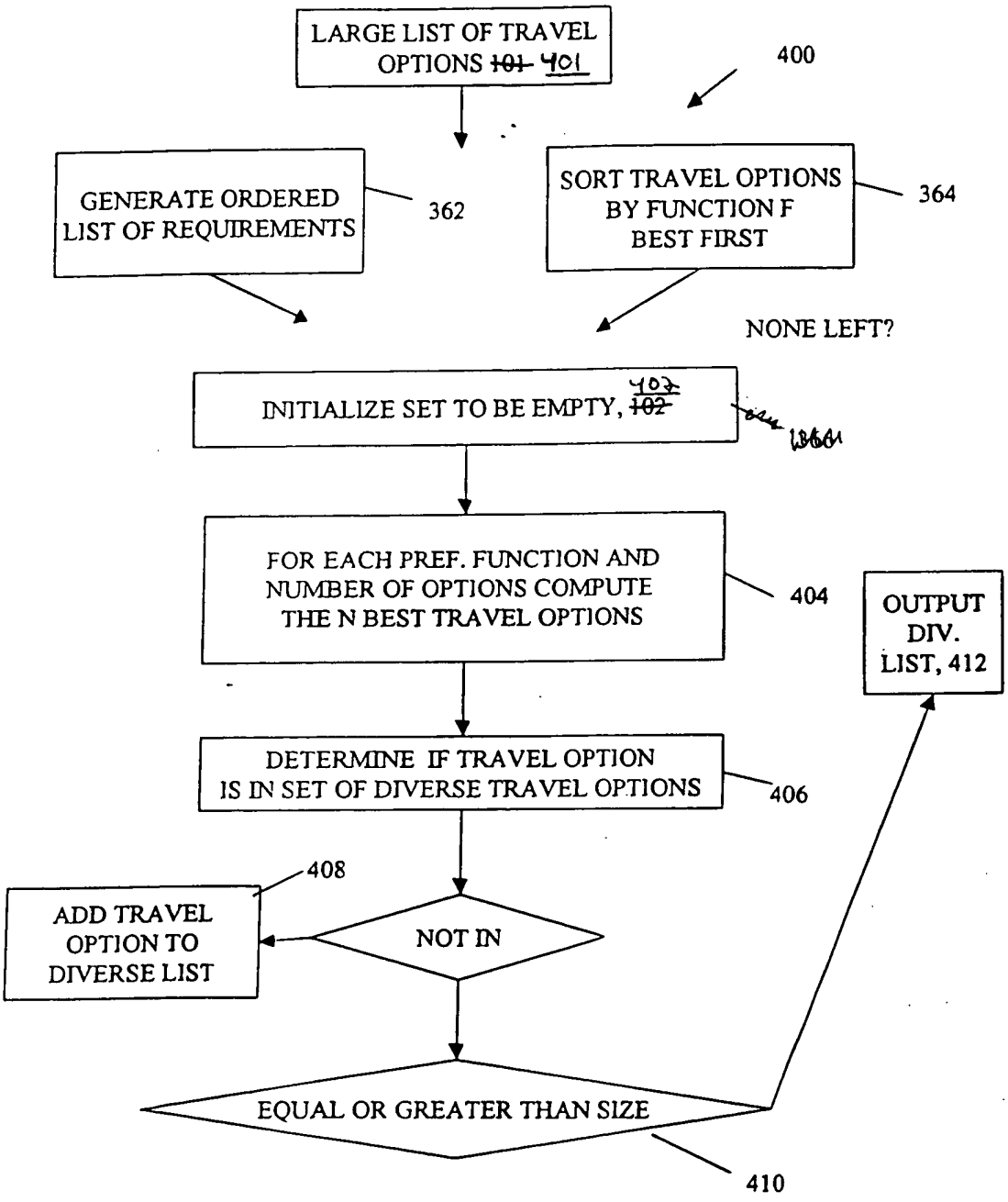
LARGE LIST OF TRAVEL OPTIONS ~~101~~ 401
400
GENERATE ORDERED LIST OF REQUIREMENTS
362
SORT TRAVEL OPTIONS BY FUNCTION F BEST FIRST
364
NONE LEFT?
INITIALIZE SET TO BE EMPTY, ~~102~~ 403
FOR EACH PREF. FUNCTION AND NUMBER OF OPTIONS COMPUTE THE N BEST TRAVEL OPTIONS
404
OUTPUT DIV. LIST, 412
DETERMINE IF TRAVEL OPTION IS IN SET OF DIVERSE TRAVEL OPTIONS
406
408
ADD TRAVEL OPTION TO DIVERSE LIST
NOT IN
EQUAL OR GREATER THAN SIZE
410

FIG. 12